Byzantine **Ordered** Consensus without Byzantine Oligarchy

Yunhao Zhang[†], Srinath Setty[*], Qi Chen[*], Lidong Zhou[*] and Lorenzo Alvisi[†]

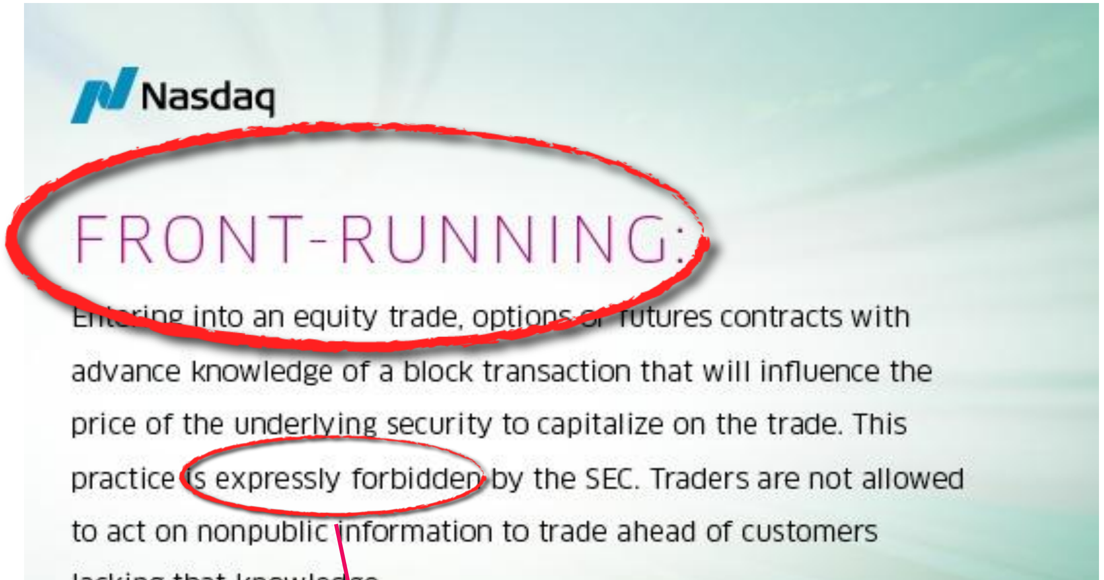
[†]*Cornell University* [*]*Microsoft Research*

ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED

Order manipulation is a scourge



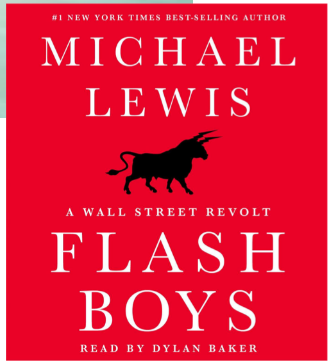
Nasdaq

FRONT-RUNNING:

Entering into an equity trade, options or futures contracts with advance knowledge of a block transaction that will influence the price of the underlying security to capitalize on the trade. This practice is expressly forbidden by the SEC. Traders are not allowed to act on nonpublic information to trade ahead of customers lacking that knowledge.

Expressly forbidden…

…but keeps happening!



#1 NEW YORK TIMES BEST-SELLING AUTHOR

MICHAEL LEWIS

A WALL STREET REVOLT

FLASH BOYS

READ BY DYLAN BAKER

Bots have reaped from unsuspecting parties over **$6M** in Ethereum!

Permissioned blockchains are vulnerable

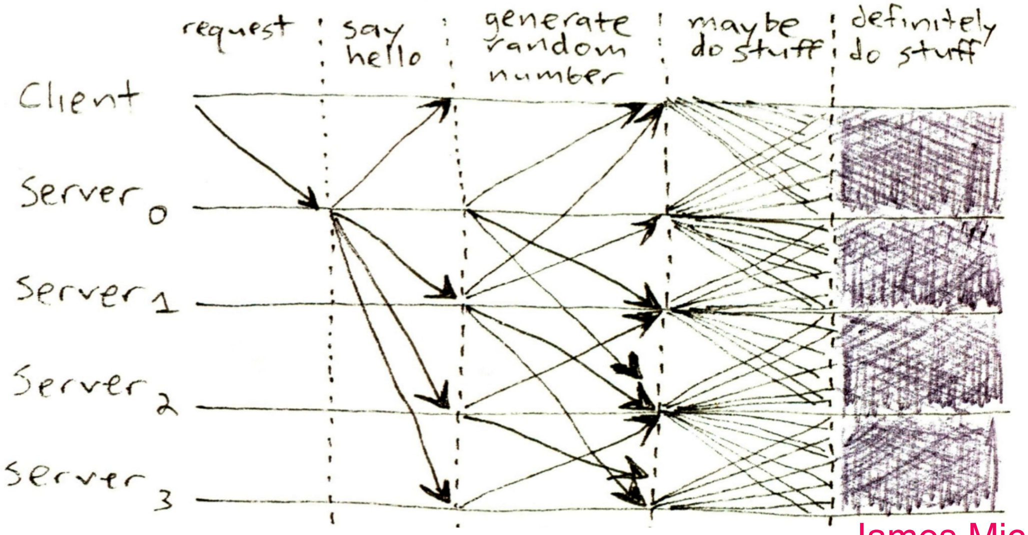
libra

HYPERLEDGER

- Promise trustworthy trading platforms.
- Rely on BFT State Machine Replication…
  - …and that's where the vulnerability lies

Oh no! BFT!

The issue is **NOT** with this →

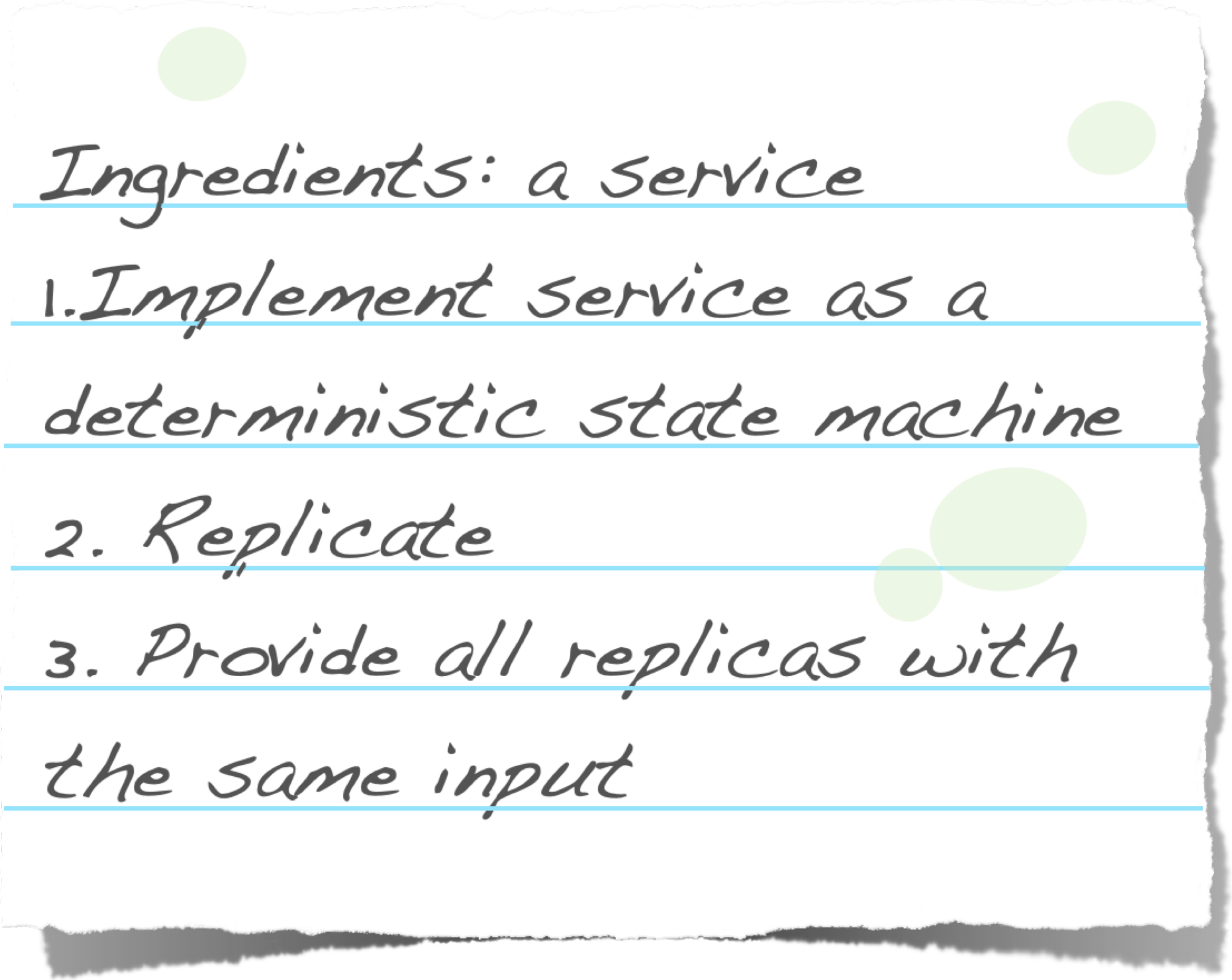


James Mickens ™

It's **worse!**

It affects *correctness specification* of state machine replication.

State Machine Replication



Ingredients: a service

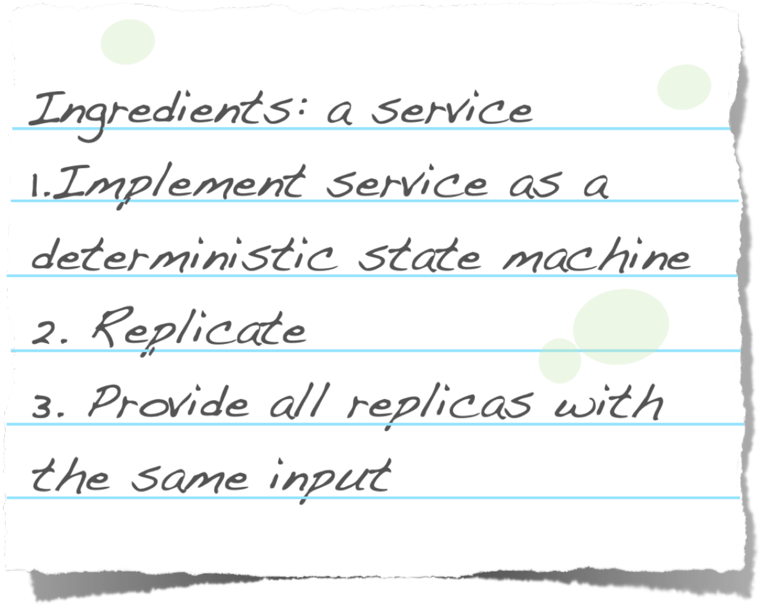
1. Implement service as a deterministic state machine
2. Replicate
3. Provide all replicas with the same input

Safety: The ledgers of correct replicas hold the same sequence of commands.

Liveness: Commands from correct clients eventually appear in the ledgers of all correct replicas.

+ BFT: S&L hold even when faulty nodes are Byzantine.

The crux



Ingredients: a service

1.Implement service as a deterministic state machine

2. Replicate

3. Provide all replicas with the same input

libra

HYPERLEDGER

When it’s about fault-tolerance

order does not matter

When it’s about financial transactions

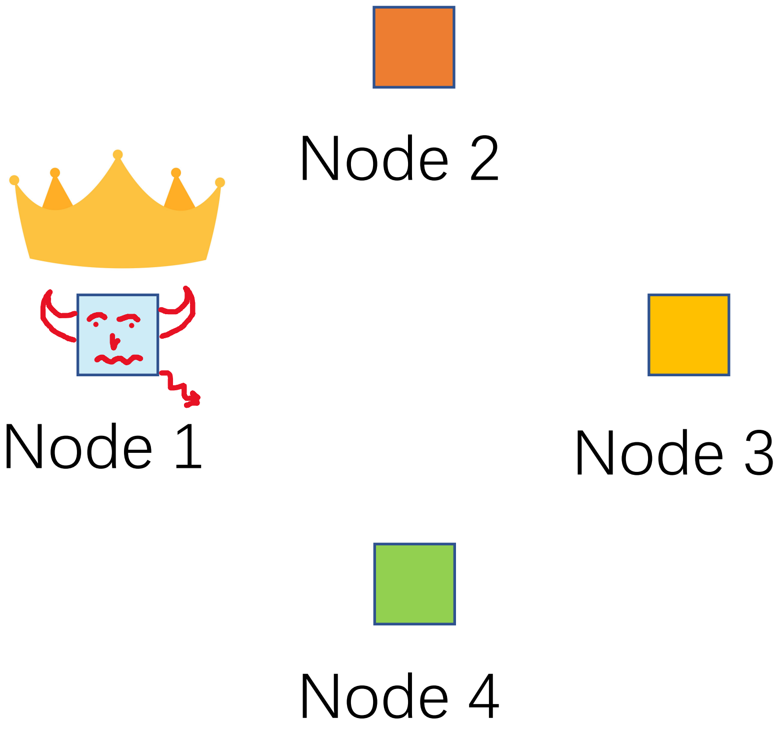
order matters!

Following the leader?

Most BFT RSM protocols are leader-based.

Leader has full control over the ledger's order.

Bad if leader is Byzantine.

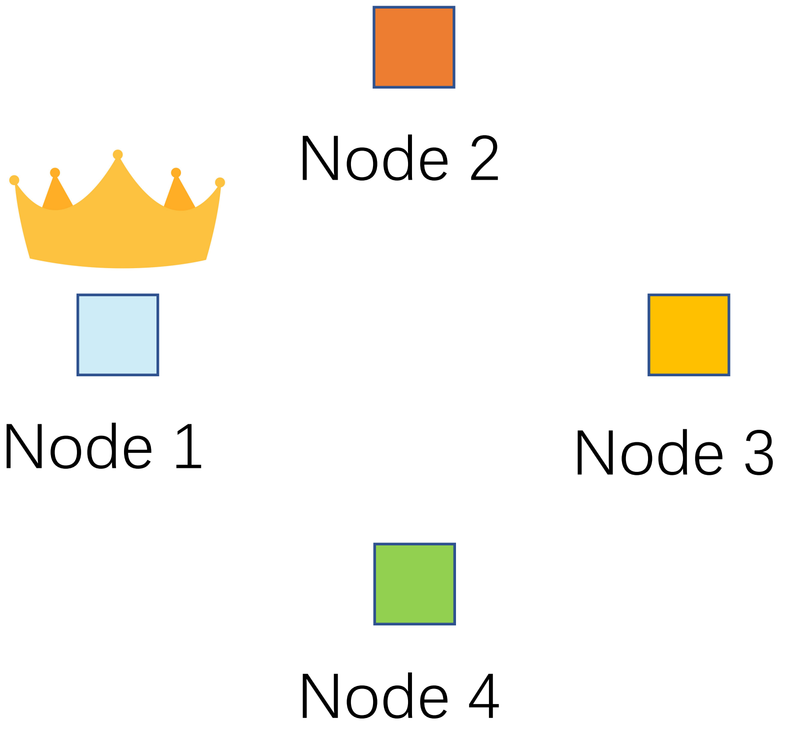


Rotating leaders

libra

Yet…

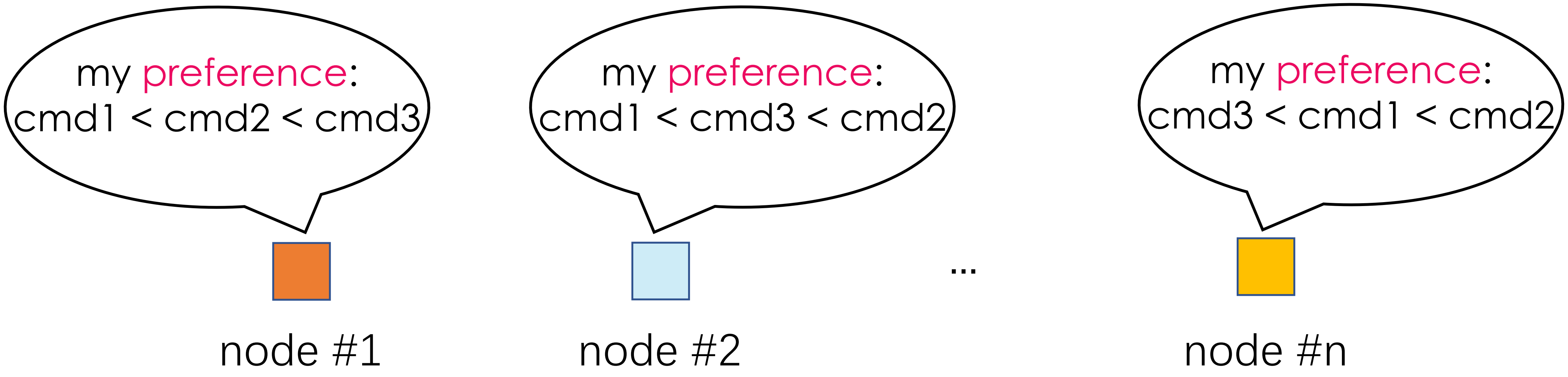
- Each leader still controls order of commands in its batch.
- No way to express correctness conditions on resulting total order.



Our main contributions

- ***Contribution #1***: Expand the BFT SMR specification
  - To express ordering requirements rigorously and define ordered consensus
- ***Contribution #2:*** Chart the boundaries of Byzantine influence
  - To understand which requirements can and cannot be enforced
- ***Contribution #3***: Articulate a new architecture for BFT SMR
  - To enforce ordered consensus
- ***Contribution #4***: Design, implement, and evaluate Pompē
  - To demonstrate systems based on ordered consensus are practical

#1: Byzantine ordered consensus



my preference:
cmd1 < cmd2 < cmd3

my preference:
cmd1 < cmd3 < cmd2

my preference:
cmd3 < cmd1 < cmd2

node #1

node #2

…

node #n

Example: ordering unanimity

if all correct nodes prefer cmd1 < cmd2,

then cmd1 < cmd2 in the output ledger.

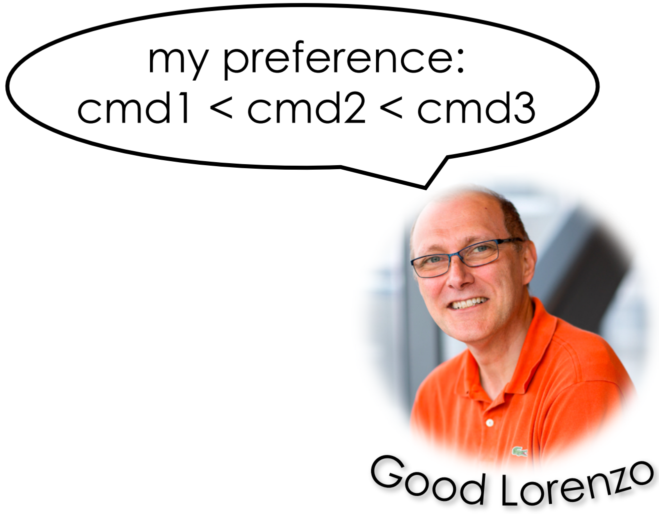
Impossibility of unanimity

🟦 Node 1	cmd1 < cmd2 < cmd3 < cmd4
🟧 Node 2	cmd2 < cmd3 < cmd4 < cmd1
🟨 Node 3	cmd3 < cmd4 < cmd1 < cmd2
🟩 Node 4	cmd4 < cmd1 < cmd2 < cmd3

#2 Understanding the limits of Byzantine sway

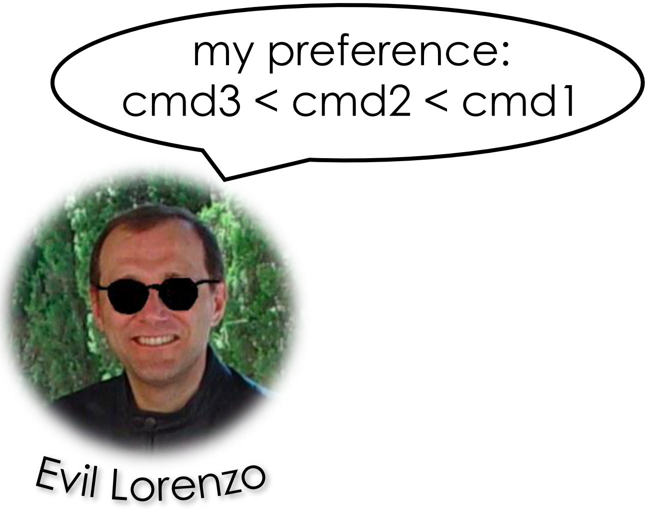
- **The good news:** We can prevent Byzantine nodes from dictating the final total order.

- **The bad news** : We cannot fully eliminate Byzantine influence.



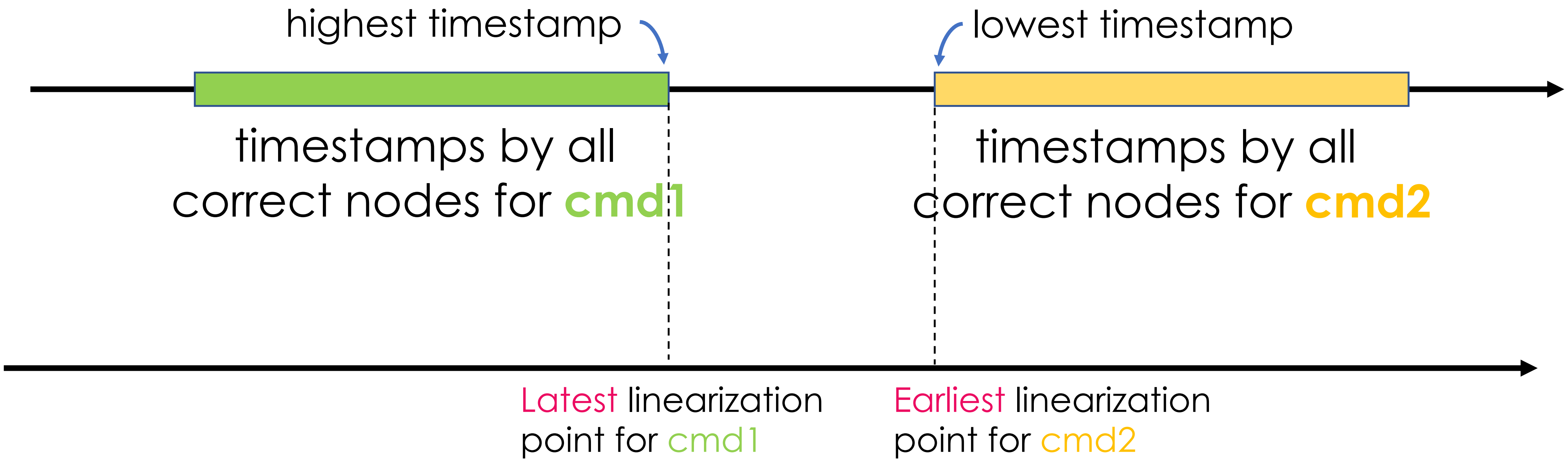
cannot distinguish
correct from Byzantine

but can still express
useful and natural
ordering guarantees



Ordering Linearizability

- Expresses ordering preferences as timestamps.



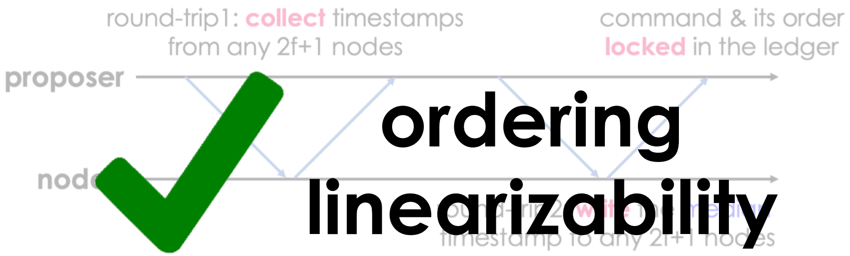
highest timestamp
lowest timestamp
timestamps by all correct nodes for cmd1
timestamps by all correct nodes for cmd2
Latest linearization point for cmd1
Earliest linearization point for cmd2

#3: A new architecture for BFT SMR

- Separate Ordering from Consensus
  - Ordering phase decides the relative order of commands.
    - Prevents Byzantine nodes from controlling ordering.
  - Consensus phase periodically decides a prefix of the ledger.
    - Can preserve performance benefits of leader-based consensus.

#4: Pompē: order-linearizable SMR

two variants of Pompē


round-trip1: collect timestamps from any 2f+1 nodes
command & its order locked in the ledger
proposer
node
round-trip2: write the median timestamp to any 2f+1 nodes

ordering linearizability

Pompē-HS: ≋libra (HotStuff)

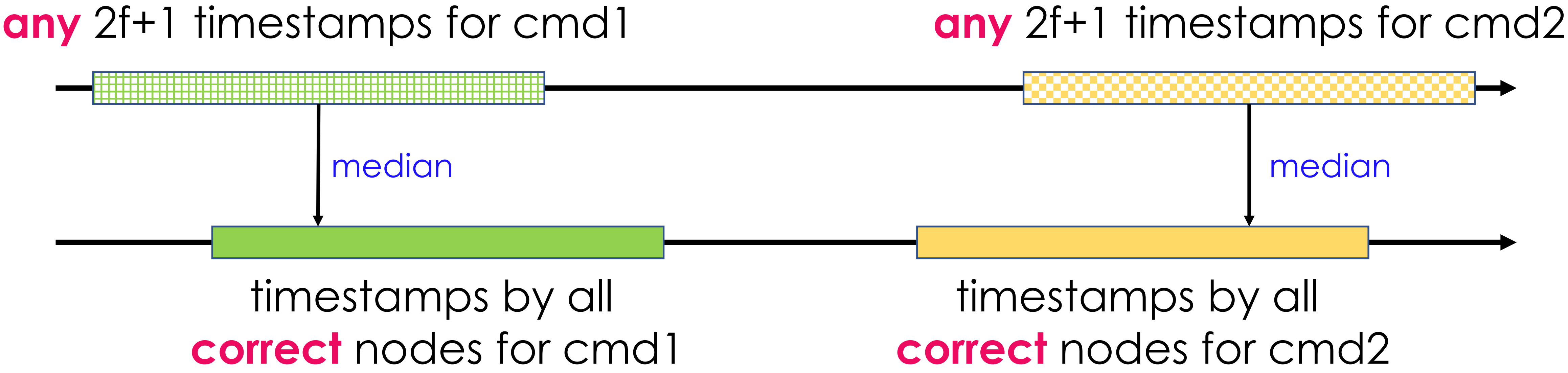
Pompē-C: CONCORD

same ordering phase

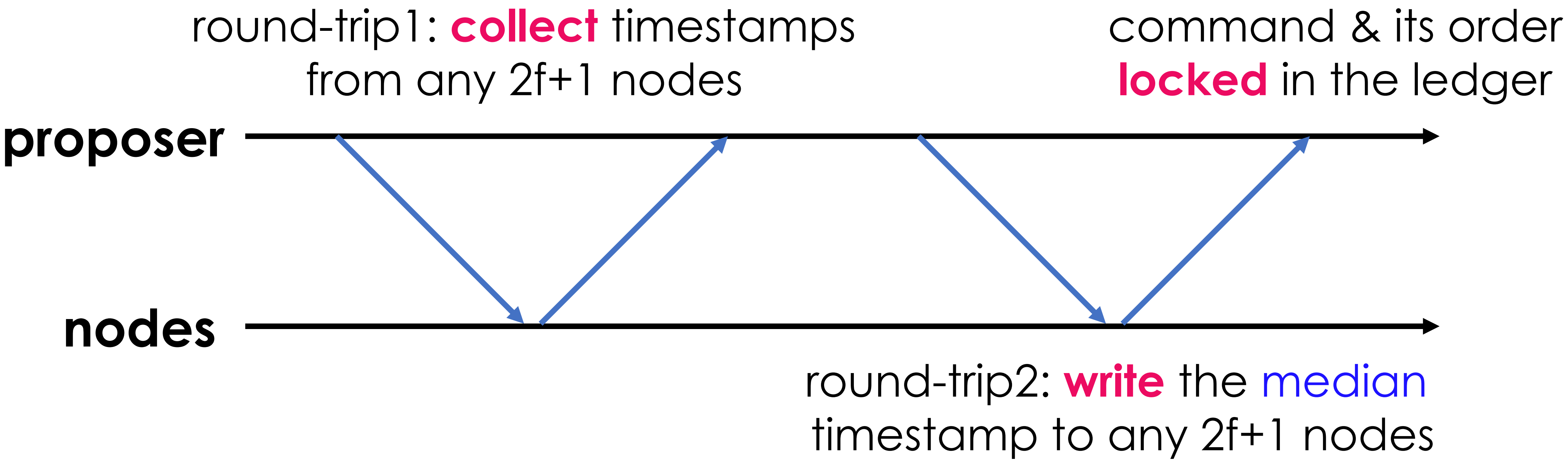
different consensus phase

Building a Byzantine-tolerant timestamp

- Assume 3f+1 nodes, f Byzantine


any 2f+1 timestamps for cmd1
any 2f+1 timestamps for cmd2
median
median
timestamps by all **correct** nodes for cmd1
timestamps by all **correct** nodes for cmd2

Locking the median timestamp



round-trip1: **collect** timestamps from any 2f+1 nodes

command & its order **locked** in the ledger

proposer

nodes

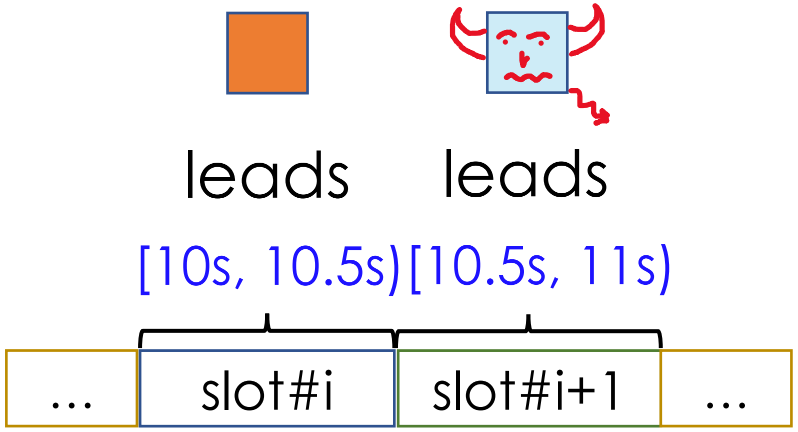
round-trip2: **write** the median timestamp to any 2f+1 nodes

Consensus phase in Pompē

- Associates each consensus slot with a time interval.
- Waits until commands issued in current time interval are locked.
- Collects newly locked commands & their timestamps.
- Uses any SMR protocol to add these commands to the ledger according to their timestamps.

Safe batching in consensus phase

Pompē



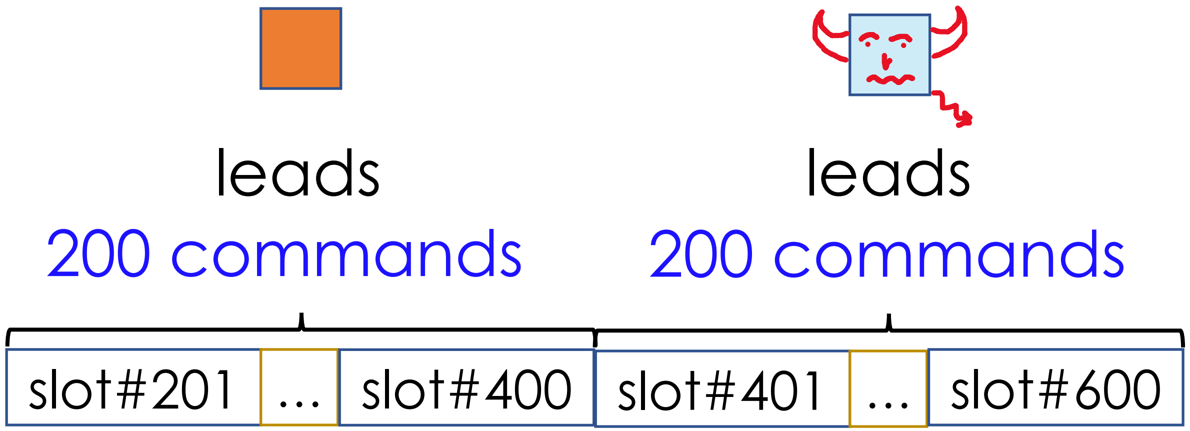
leads leads

[10s, 10.5s) [10.5s, 11s)

…	slot#i	slot#i+1	…

order free from
Byzantine leader's control

state-of-the-art



leads leads

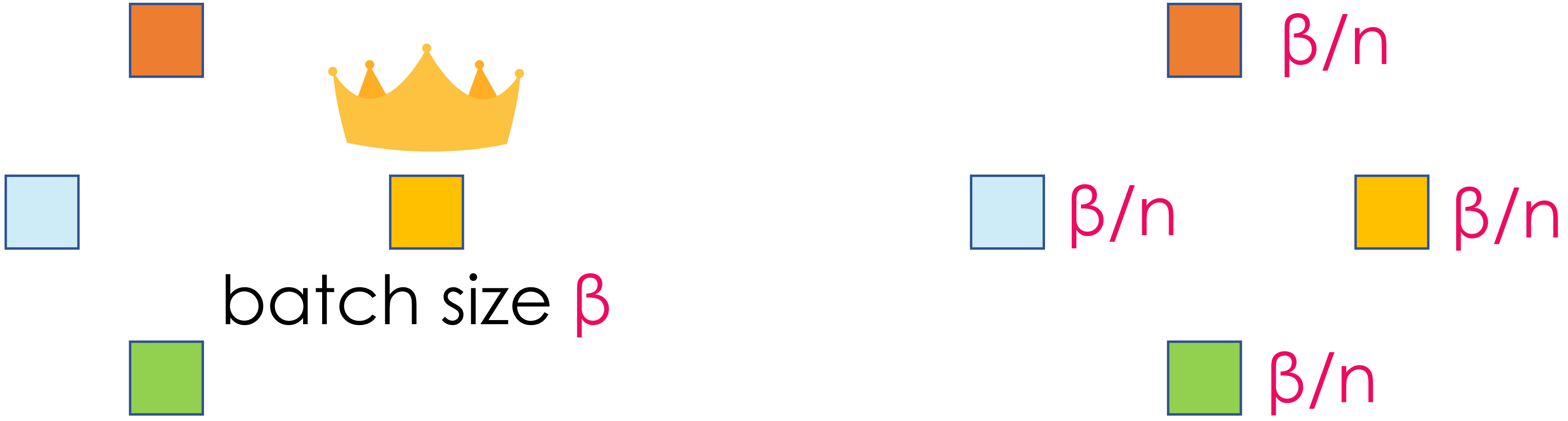
200 commands 200 commands

slot#201	…	slot#400	slot#401	…	slot#600

order subject to
Byzantine leader's control

Batching during the ordering phase

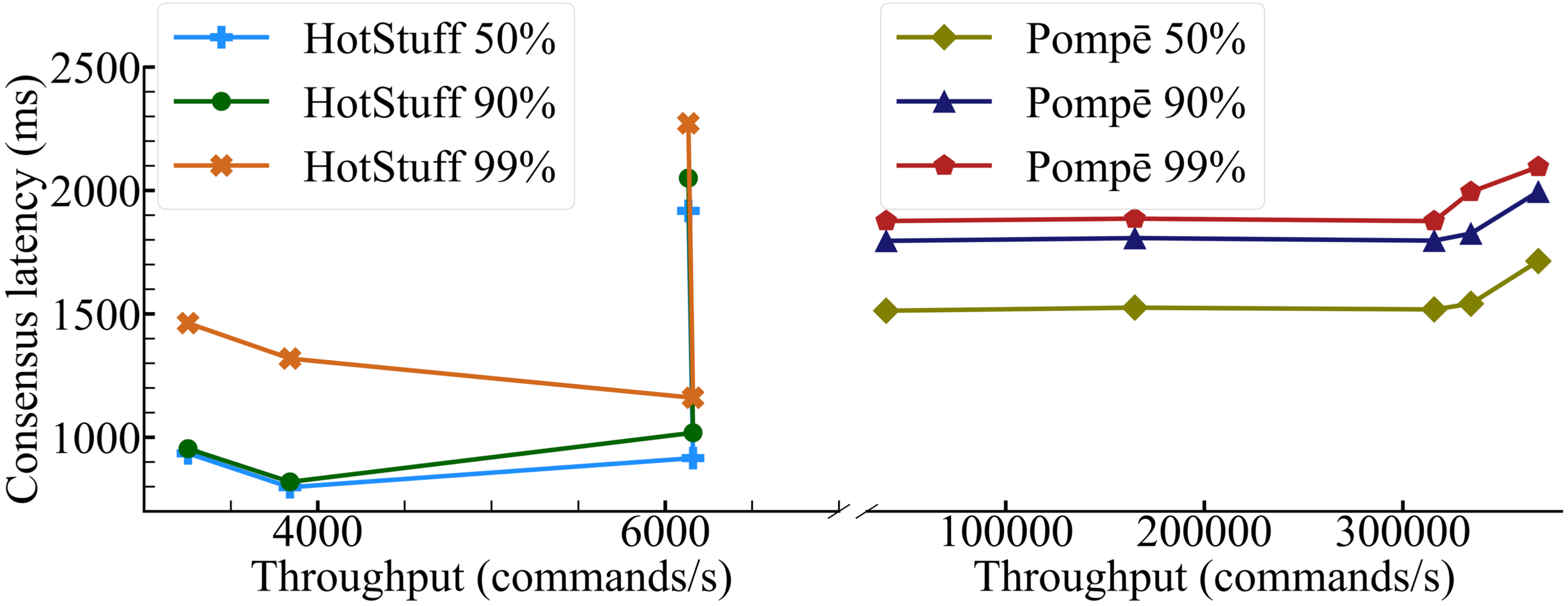
- A single timestamp to a batch from the same node
- For the purposes of evaluation:



Baseline

Pompē

Pompē vs HotStuff: 4 geo-distributed nodes



Conclusion

- There is a fundamental gap between the SMR correctness spec and the threat from order manipulation in blockchains.
- We introduce a new primitive, ordered consensus, to allow rigorous expression and efficient enforcement of ordering requirements.
- We design a modular architecture for ordered consensus and built Pompē which enforces ordering linearizability with performance comparable to state-of-the-art systems.

Thanks for listening! Any questions?

- There is a fundamental gap between the SMR correctness spec and the threat from order manipulation in blockchains.
- We introduce a new primitive, ordered consensus, to allow rigorous expression and efficient enforcement of ordering requirements.
- We design a modular architecture for ordered consensus and built Pompē which enforces ordering linearizability with performance comparable to state-of-the-art.

For further questions,
feel free to contact Yunhao (yz2327@cornell.edu).